Traits

Traits

Similar to an interface or abstract class

Traits

Similar to an interface or abstract class

Add a definition to a structure

Traits

Similar to an interface or abstract class

Add a definition to a structure

```
trait Name {
   fn must_implement(&self) -> i32;
   fn do_action(&self) { ... }
   fn do_non_instance_action() { ... }
}
```

Traits

Similar to an interface or abstract class

Add a definition to a structure

```
trait Name {
   fn must_implement(&self) -> i32;
   fn do_action(&self) { ... }
   fn do_non_instance_action() { ... }
}
```

Can have definition only or default implementation

Traits

Similar to an interface or abstract class

Add a definition to a structure

```
trait Name {
   fn must_implement(&self) -> i32;
   fn do_action(&self) { ... }
   fn do_non_instance_action() { ... }
}
```

Can have definition only or default implementation

Can have instance and non-instance action

Traits

Implement a trait

```
impl Name for Person {
   fn must_implement(&self) -> { 42 }
}
```

Traits

Implement a trait

```
impl Name for Person {
  fn must_implement(&self) -> { 42 }
}
```

Can provide a constructor

```
trait Name {
  fn new(name: &str) -> Self;
}
```

Traits

Implement a trait

```
impl Name for Person {
  fn must_implement(&self) -> { 42 }
  fn new(name: &str) -> Person {
    Person{name: name}
  }
}
```

Can provide a constructor

```
trait Name {
  fn new(name: &str) -> Self;
}
```

Traits

Implement a trait

```
impl Name for Person {
  fn must_implement(&self) -> { 42 }
  fn new(name: &str) -> Person {
    Person{name: name}
  }
}
```

Can provide a constructor

```
trait Name {
  fn new(name: &str) -> Self;
}
```

```
let john = Person::new(“John”);
```